https://ntrs.nasa.gov/search.jsp?R=20170001294 2020-05-09T09:43:08+00:00Z

Functional Fault Model Development Process to Support Design Analysis and Operational Assessment

Kevin J. Melcher[1]
NASA Glenn Research Center, Cleveland, Ohio, 44135, USA

William A. Maul[2]
Vantage Partners, LLC, Brook Park, Ohio, 44142, USA

and

Joseph A. Hemminger[3]
Zin Technologies, Inc., Brook Park, Ohio, 44142, USA

A functional fault model (FFM) is an abstract representation of the failure space of a given system. As such, it simulates the propagation of failure effects along paths between the origin of the system failure modes and points within the system capable of observing the failure effects. As a result, FFMs may be used to diagnose the presence of failures in the modeled system. FFMs necessarily contain a significant amount of information about the design, operations, and failure modes and effects. One of the important benefits of FFMs is that they may be qualitative, rather than quantitative and, as a result, may be implemented early in the design process when there is more potential to positively impact the system design. FFMs may therefore be developed and matured throughout the monitored system's design process and may subsequently be used to provide real-time diagnostic assessments that support system operations. This paper provides an overview of a generalized NASA process that is being used to develop and apply FFMs. FFM technology has been evolving for more than 25 years. The FFM development process presented in this paper was refined during NASA's Ares I, Space Launch System, and Ground Systems Development and Operations programs (i.e., from about 2007 to the present). Process refinement took place as new modeling, analysis, and verification tools were created to enhance FFM capabilities. In this paper, standard elements of a model development process (i.e., knowledge acquisition, conceptual design, implementation & verification, and application) are described within the context of FFMs. Further, newer tools and analytical capabilities that may benefit the broader systems engineering process are identified and briefly described. The discussion is intended as a high-level guide for future FFM modelers.

Nomenclature

Acronym	Description
AGSM	Advanced Ground Systems Maintenance
CSV	Comma Separated Variable
ETA	Extended Testability Analysis
FFM	Functional Fault Model
FFMs	Functional Fault Models
FMEA	Failure Modes and Effects Analysis
GEMINI	GEneric Model INstantIator
GUI	Graphical User Interface

[1] Team Lead, Systems Health Management Methods for Space Exploration, Intelligent Control and Autonomy Branch, M.S. 77-1, AIAA Senior Member.
[2] Aerospace Engineer, M.S. VPL-3, AIAA Senior Member.
[3] Systems Engineer, M.S. VPL-3, AIAA Senior Member.

Acronym	Description
ID	Identifier
IDU	Integrated Health Management Demonstration for Universal Propellant Servicing System
ISS	International Space Station
LRU	Line Replaceable Unit
QSI	Qualtech Systems Inc.
S&MA	Safety and Mission Assurance
SBIR	Small Business Innovative Research
SBS	System Breakdown Structure
SLS	Space Launch System
SME	Subject Matter Expert
TDF	Test Description File
TEAMS	Testability Engineering And Maintenance System
VERA	VERification Analysis
VV&A	Verification, Validation, and Accreditation

I. Introduction

Functional fault models (FFMs) are abstract failure space representations that define the effect propagation paths of critical failure modes associated with a given system. These models embody information typically found in a Failure Modes and Effects (FMEA) analysis. However, because they define effect propagation paths as well as the failure modes and detection mechanisms, an FFM can provide insight into the safety and reliability characteristics of the system design. Initially an analytical tool supporting the system design, a FFM can be developed into a powerful, real-time diagnostic tool supporting system operations. FFMs can be developed using qualitative, as well as quantitative, relationships to describe system behavior in its failure space. One advantage of qualitative FFMs is that they may be developed early in the system design process when hard numbers (i.e., quantitative data) are not well-defined or may be evolving with the design[1]. It is during this portion of the design that FFMs are more likely to positively impact the design by identifying unmet requirements for off-nominal operation when changes to the design are less costly.

In the late-1980s and early 1990s, a number of researchers were focused on investigating the use of approaches based in testability analysis to provide more computationally efficient fault diagnosis (i.e., detection and isolation) techniques. During this time, Padalkar[2], et al, proposed a graph-based technique for real-time fault diagnostics. Subsequently, Pattipati and Alexandridis[3] integrated information theory with heuristic AND/OR graph search methods (i.e., directed graph) to perform diagnosis of single faults on electronic and electro-mechanical systems with large numbers of failure modes. In 1994, the methods developed by Pattipati, et al, were subsequently presented in the form of an X-windows based software tool, the Testability Engineering and Maintenance System (TEAMS)[4] from Qualtech Systems, Inc. (QSI). TEAMS was presented as a multi-signal, directed-graph modeling approach with a graphical user interface that allowed its diagnostic models to closely resemble hierarchical system schematics[5].

NASA researchers became interested in graph-based fault diagnosis[6-8] around the same time that Pattipati, et al, were conducting their work. In 1993, Iverson and Patterson-Hine[9] of NASA Ames Research Center reported on the development of a new algorithm to improve the performance of directed graph (digraph) based searches that reduced the solution time from hours or days to minutes. Since these methods showed promise, NASA solicited help via the Small Business Innovative Research (SBIR) program to further develop directed graph-based fault diagnosis capabilities. QSI responded to that solicitation and won its first of a long series of NASA SBIR awards in 1994. This and subsequent SBIRs focused on maturing the development of multi-signal, digraph-based diagnostic approaches embodied by QSI's TEAMS software and on demonstrating the utility of that software for systems of particular interest to NASA.

In the more than two decades following their initial collaboration, QSI and NASA have been advancing the utility of TEAMS-based diagnostic models – currently known as FFMs under NASA's Space Launch System (SLS) Program – and the application of these types of models to complex systems. The broad applicability of FFMs is highlighted by the variety of systems to which they have been applied by NASA: UH-60 helicopter[10], the International Space Station (ISS)[11], the Space Shuttle[12], launch & ground systems for the Ares I launch vehicle[13-17], and ground systems software for the Orion Multi-purpose Crew Vehicle[18]. It was demonstrated during these activities that FFMs may be evolved with the system design to the point that an FFM that was created to support early design verification could be evolved and used to provide a real-time diagnostic assessment of the system. Patterson-Hine[10] noted that this design-to-operation development and implementation approach increases the confidence of the stakeholder in the diagnostic capability of the FFM. Further, efforts by Deb showed how FFMs could be used to diagnose faults in safety critical, highly-connected, and broadly-distributed networked systems like the ISS 1553 bus[11] and in Space Shuttle wiring systems[12]. They have also been demonstrated in real-time launch vehicle diagnostic assessments[13] and for advanced ground-based launch system processing[14] under NASA's Constellation Program.

The most recent efforts have included the development of cross-program FFM modeling conventions[19], the expansion of analytical tools[15-17,20], and library-based modeling approaches[21]. The new analytical tools provide more efficient verification of FFMs and their associated fault management requirements while library-based modeling approaches can result in reduced model development time. To accomplish these goals, FFMs must maintain a level of consistency during development that enables efficient revision and integration of the models with external processing and reporting software tools, real-time diagnostic architectures, and other FFMs. Under the Ground System Development and Operations program, the Space Launch System Program and the Orion Program, a set of cross-program FFM modeling conventions[19] were established to address these issues. The utility of these new capabilities is highlighted by the fact that several of them have been integrated by QSI into the commercial version of their TEAMS software.

As the utility of FFMs expand and new modeling and analysis capabilities become available, the model development process must obviously evolve and adapt to take advantage of the new capabilities. The purpose of this paper is to describe a process for developing FFMs that has evolved in recent years with the result that FFMs can be developed more efficiently and can provide broader support for failure space design, analysis, and operational assessments. The discussion is organized as follows. First, an overview of the process is presented which includes a process diagram to facilitate the discussion. Then each phase of the process diagram is described in detail to facilitate assimilation of the process. As the process is described, the paper will highlight model input data requirements and the impact and benefit of current modeling conventions and software tools on model development and analysis processes. In the final sections of the report, concluding remarks and acknowledgements are given.

II. Description of the FFM Development Process

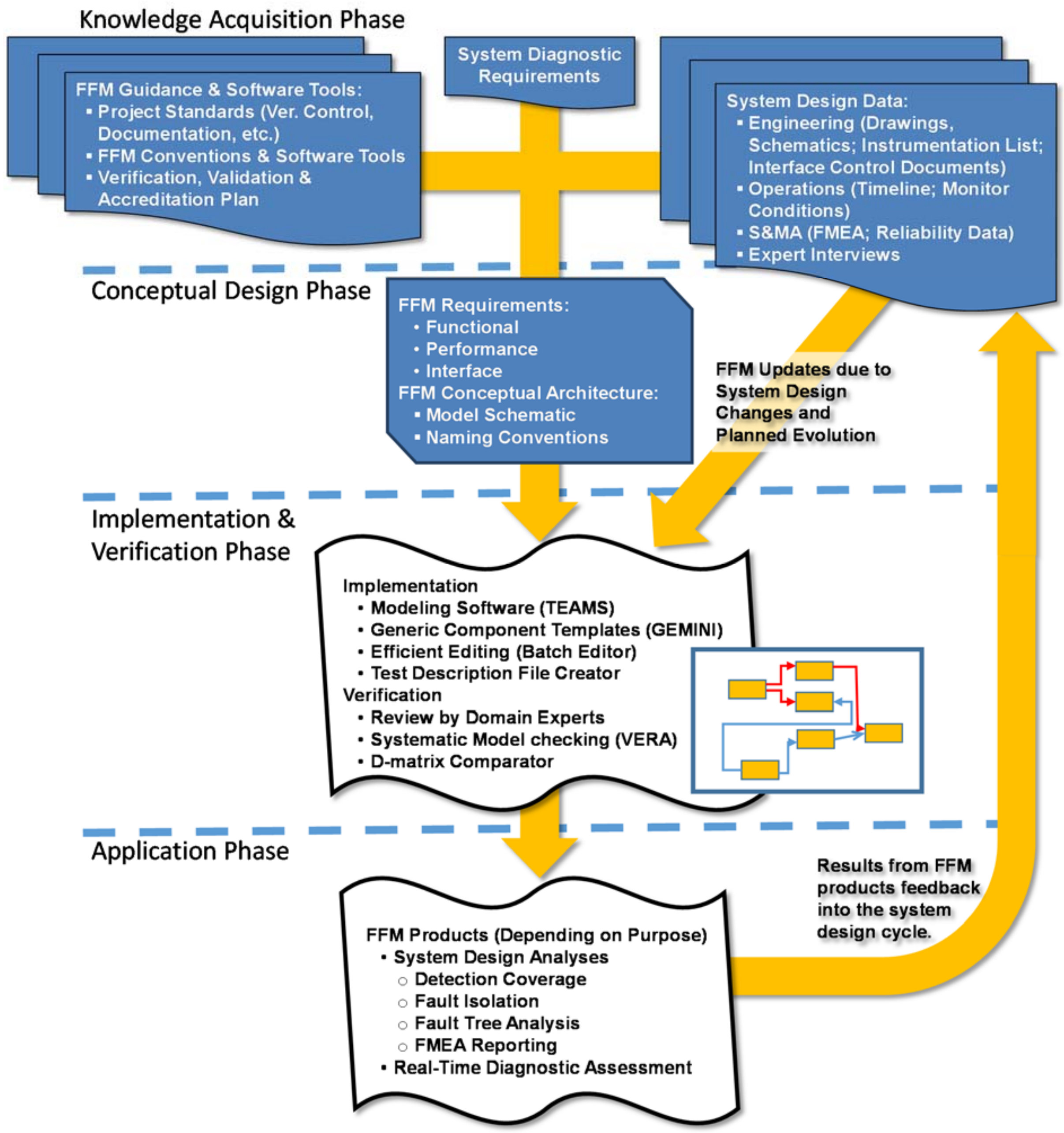
The FFM development process described in this paper has four phases: knowledge acquisition, conceptual design, implementation & verification, and application. As shown in Fig. 1, this process flows system diagnostic requirements, system design data, and current FFM modeling practices and capabilities into the conceptual design. The conceptual design then flows through model implementation and verification into resulting analysis products whose results, ideally, feed back into systems engineering and design processes. The four phases of development will be discussed in more detail throughout the remainder of this section. At a high level, this process does not seem that different from historical modeling approaches. However, as the paper delves into the lower-levels of the process, a host of newer tools and capabilities will be discussed that have been shown to improve the efficiency and utility of one or more of the four phases of FFM development and, consequently, the FFM development process as a whole.

A. Knowledge Acquisition

The FFM process starts with a traditional knowledge acquisition phase. During this phase, the modeler must first identify system diagnostic requirements that will inform and bound all phases of the FFM development process. For mature designs, these requirements typically come from higher-level design requirements documents. However, if the design is immature, the system diagnostic requirements may not exist at a low enough level to be useful for FFM development. In that case, the modeler may need to work with the system designer(s) to define an initial set of system diagnostic requirements.

Next, the modelers should begin assembling the tools, processes, and information required to efficiently and accurately create and utilize FFMs. Elements of this phase include the creation of knowledge, tools, and FFM depositories as well as the gathering of development software, FFM guidance documents, and system design data to populate the depositories. The effective organization of, and usage plans for, this knowledge and capability early in the process can minimize model data errors and save significant resources later in the project.

The selection, creation, and well-planned organization of knowledge depositories is an important first step in collecting this information. Content and Configuration Management guidelines for each depository help to make user access efficient and to minimize documentation errors. Version control software can be utilized to configuration manage documents, software, and models. Examples of separate depositories that can be created for different knowledge categories include archives for: (1) source/official documents; (2) working documents; (3) official documents; (4) software tools for development and analysis; (5) FFMs. For information associated with the system being modeled, an integrated System Breakdown Structure (SBS) database may provide an efficient means of coordinating data for use by multiple FFM modelers who are individually developing various subsystem models.



Knowledge Acquisition Phase
FFM Guidance & Software Tools:
▪ Project Standards (Ver. Control, Documentation, etc.)
▪ FFM Conventions & Software Tools
▪ Verification, Validation & Accreditation Plan
System Diagnostic Requirements
System Design Data:
▪ Engineering (Drawings, Schematics; Instrumentation List; Interface Control Documents)
▪ Operations (Timeline; Monitor Conditions)
▪ S&MA (FMEA; Reliability Data)
▪ Expert Interviews
Conceptual Design Phase
FFM Requirements:
• Functional
• Performance
• Interface
FFM Conceptual Architecture:
▪ Model Schematic
▪ Naming Conventions
FFM Updates due to System Design Changes and Planned Evolution
Implementation & Verification Phase
Implementation
• Modeling Software (TEAMS)
• Generic Component Templates (GEMINI)
• Efficient Editing (Batch Editor)
• Test Description File Creator
Verification
• Review by Domain Experts
• Systematic Model checking (VERA)
• D-matrix Comparator
Application Phase
Results from FFM products feedback into the system design cycle.
FFM Products (Depending on Purpose)
• System Design Analyses
○ Detection Coverage
○ Fault Isolation
○ Fault Tree Analysis
○ FMEA Reporting
• Real-Time Diagnostic Assessment

Figure 1. Functional fault model development process

System Design Data: A key next step in the knowledge acquisition phase is the identification of system design data relevant to the development of a specific FFM. These data are typically comprised of engineering data, operations data, and Safety and Mission Assurance (S&MA) data, as well as data from interviews with subject matter experts (SMEs). Identification of SMEs early in the FFM development process, regular planned interactions with them during FFM development, and the documentation of their input and feedback, are very important to the FFM development process.

Engineering data includes system architecture diagrams, hardware drawings and schematics which detail the configuration of the system. It also includes unique and succinct element identifiers (IDs) that can be traced back to the drawings and schematics. These IDs can provide unique links between hardware and the FFM model, links that are critical for model verification and for reporting the results of diagnostic assessments. ID information is typically obtained from: parts lists, component data sheets, instrumentation lists, and command and data signals. Engineering data also includes interface control documents which provide details of the interconnections between system elements. This collection of engineering documents guides the creation of FFM component elements, test points, and their inter-connectivity.

Operations data includes documentation describing the concept of operations, operating procedures, mission timelines, and monitored conditions. These documents are used to define operating states of elements in a system for a given mission and/or phases within each mission. Failure propagation paths within the FFM can be configured using various component element states (e.g., valve open/valve closed) to appropriately redirect propagation paths for a specific mode of operation. A defined set of monitored conditions typically result from a flow down of the system diagnostic requirements associated with loss of mission, loss of crew and accepted levels of risk.

S&MA data guides the definition of system failure modes and the effects propagated that may be represented within the FFM. S&MA documentation includes FMEAs, Hazard Analyses and reliability analyses which are used to define the failure mode credibility thereby scoping the FFM. FMEAs contain a substantial amount of data utilized within the FFMs. These documents utilize the same engineering and operations data as the FFMs and provide crucial failure mode definition that is transferred into the FFM. FMEAs also provide unique identifiers for the failure modes that will enable traceability between the FMEAs and the FFM. Since FFM development and support software can in turn generate FMEA reports, there is potential in the future that the FFMs could be developed initially and used to generate the FMEA products.

<u>FFM Guidance & Software Tools</u>: The final key step in the knowledge acquisition phase is the identification and acquisition of information and software tools that are to be used for the FFM development process from systems engineering and modeling perspectives, as opposed to a system design perspective. Examples of information that may be required are project standards and directives, established modeling conventions, modeling software and NASA-developed FFM software tools (includes user guides and training materials), and verification, validation, and accreditation (VV&A) plans. An understanding of project standards by the modeler will ensure that FFM products meet requirements for version control and model documentation. Established modeling conventions enhance reporting, verification, and the integration of FFMs and are required for the utilization of recently developed FFM support tools. An example of modeling software utilized by NASA is QSI's TEAMS software. TEAMS's front-end graphical user interface (GUI) allows a user to create a FFM that is architecturally similar to hardware and software schematics of the system being modeled. FFM software tools were developed by NASA to augment the capabilities of TEAMS software. These tools may be identified as model implementation tools, VV&A tools, or analysis reporting tools, and will be reported in more detail in subsequent sections.

Finally, it should be noted that knowledge acquisition typically occurs throughout the FFM development process. Consequently, the modeler should proceed to the Conceptual Design Phase when sufficient knowledge has been collected.

B. Conceptual Design

As the domain knowledge is accumulated and diagnostic requirements levied on the FFM, the FFM developer may begin to develop the model conceptual design. There are three primary goals of this model development phase. The first goal is to define FFM requirements that will guide model implementation. The second goal is to define a conceptual architecture that will allow the resulting FFM to meet those requirements. And the third goal is to define the model constraints that provide scope and bound the FFM.

1. FFM Requirements Definition

To meet this goal, the system diagnostic requirements identified as part of the Knowledge Acquisition Phase should be flowed down by the modeler into requirements for the FFM. FFM models may be used to support design development, analytical verification of system diagnostic requirements, and real-time diagnosis during operations. To provide this support, FFM requirements should include functional requirements, performance requirements, and interface requirements. Functional requirements define the failure modes the model must detect, the mechanisms available to detect them, and the level of isolation (e.g., component, subsystem, system) required for each failure mode. Functional requirements also include the analyses the model will be required to perform. Under NASA's Constellation and SLS Programs, FFMs have been used to provide a number of new analytical approaches to support verification of system diagnostic requirements and algorithms. Examples include: the verification of fault isolation requirements for proposed line replaceable units, the verification of algorithms to detect launch commit criteria, and sensor sensitivity studies designed to assess the impact of sensor loss to diagnostic capability. These analyses and their impact on systems engineering and design processes will be discussed in Sec. II.D Application. Performance requirements describe how well the FFM must function and typically include quantitative bounds for metrics like the time required to detect a failure mode and the accuracy of the diagnoses, the latter being quantified by false positive and false negative rates. Interface requirements may, for example, specify the nature of the interface between subsystem FFMs that will eventually be integrated into a system-level FFM; or, in the case of real-time assessment, the interface between a data bus and the FFM.

2. FFM Conceptual Architecture:

The conceptual architecture may be comprised of two elements: (1) a model schematic and SBS that define the planned hierarchical structure and (2) a system operational profile that defines specific system configurations that need to be captured by the model.

The FFM development software (i.e., TEAMS) supports a hierarchical model structure wherein a given model element can contain other model elements. In this structure, the lowest model elements represent either failure modes or mapping functions that define the transition of failure effects through nominal components. This hierarchical structure allows the model developer to decompose the system structurally in much the same way systems engineers would decompose the system. For small FFM models, a flat hierarchical structure – a structure where the entire system structure can be displayed on a single level – may be sufficient. For larger models, the system may need to be decomposed into subsystems, and then into assemblies, modules, components and parts, until the required model fidelity is obtained. As an example, Fig. 2 simultaneously shows the hierarchical layers of an FFM starting at the top with the system level (i.e., basic system), moving down to the subsystem level (i.e., vector control system), and finally to the component level (i.e., power valve component). Each level encompasses more detail than the previous level.

As part of the FFM architecture, it is useful to organize information about the system elements into a tabular SBS. This structure ideally captures the breadth and depth of the decomposed system (i.e., hardware and software elements), provides a mapping of other element data that will be used in the FFM (e.g., element and failure modes named to defined conventions), and initiates an understanding of failure-propagation pathways. Here, the breadth of the model is represented by the list of components that must be modeled while the depth is represented by the various hierarchical levels. Identifying the overall structure of the FFM, both depth and breadth, is an important part of the conceptual design. Using the SBS to explicitly identify components that are to be included in the FFM, components that should be excluded from the FFM, and to define the representation of included components, will confirm the expectations of the customers and consumers of the FFM and results of associated diagnostic assessments. For large models, this decomposition of the FFM can facilitate the parsing of model development work to various members of a modeling team.

Standardizing and documenting element names prior to implementation provides for more efficient modeling. This is especially true when the development of subsystem FFMs are partitioned among different modelers with an eye to subsequent integration. Attempts at making names succinct aids in readability of both FFM diagrams and reports. One of the challenges to FFM development is that the same failure for a component can be described in numerous ways (e.g., a valve can fail open vs. fail full open, or fail part open vs. fail part closed), and the same name can be interpreted

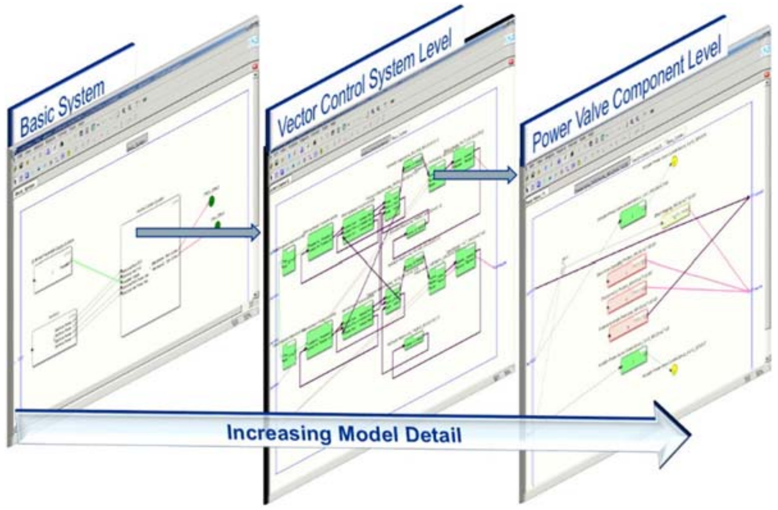


Figure 2. An example of an FFM hierarchical breakdown

differently by each developer. Providing a consistent failure mode dictionary also helps to provide consistency among the same elements in different sub-system FFMs and to catalog the same failure mode across a large FFM.

Component selection criteria should be based on requirements levied on the FFM by the program. In general the FFM should only include components that are active and could, therefore, impact model configuration or contain credible failure modes as defined by the program. It is recommended that the depth of the FFM hierarchy be somewhat consistent across the various subsystems included in the top-level system FFM. The fidelity of data in FMEAs, historically used to provide failure mode data to FFMs, can vary widely from component to component. This has an effect of biasing assessment metrics. Modelers, both individuals and teams, should be aware of this possibility and strive to balance model fidelity across components and subsystems.

The establishment of an operational profile for the system is essential in determining the complexity of the modeling required to represent the various configurations. For the FFM, a system mode defines a unique model configuration which establishes the failure mode propagation paths, which failure modes are active, and even which tests are available. In real-time applications, consideration of system mode definition must be made when test thresholds are changed due to operational profile, even though the model configuration remains the same. Some systems may have very simplistic operational modes (e.g. on or off), while other systems may have complex operational modes that are not easily represented within a single FFM.

3. Model Constraints

When laying out the conceptual design of the model, it is important that the modeler understand the constraints imposed by the FFM modeling approach described in this paper. Here, five (5) key constraints are briefly discussed for the modeler's benefit:

1) Discrete-event model;
2) Detection lag across the system;
3) Detection of multiple failures;
4) Limited propagation of failure effects;
5) Criticality and probability assignment for failure modes.

First, TEAMS-based FFMs are essentially discrete-event failure propagation models and, as such, are not time-based. Real-time applications, consequently, require data from steady or quasi-steady measurements. A wrapper code is typically used to convert real-time data from the system into discrete-event data for input to the FFM. For example, the propellant transfer from a supply tank to a vehicle tank may be considered a steady process. As such, a failure of the system may be detected testing the fluid volume rate of change in each tank against expectations that the rate of change in the supply tank would be negative and that of the vehicle tank positive.

Second, for each failure mode, the model generates a detection signature which is essentially a list of tests that detect the propagated effects of the failure. This diagnostic assessment is strict, requiring all tests to indicate detection before the failure mode is implicated. In cases where the failure effects progress over time or when system properties (e.g. pressure and temperature) lag each other, how those cases will be processed and presented, especially in a real-time application, need to be conveyed to the end-user.

Third, TEAMS FFM assessments assume that only single failure modes can occur. Combinations of failure modes cannot be assessed by the FFM unless they are completely independent of each other. An example of two independent failure modes would be a sensor failure and a component failure where the component failure mode would not be detected by the failed sensor. Unless the explicit combination of failure modes is represented in the FFM as separate and distinct failure modes, failure modes whose effects would alter or impact another failure mode, if it occurred, cannot be assessed.

Fourth, TEAMS FFMs contain failure effect propagation paths that flow in a single direction (e.g. from left to right). The modeler should carefully consider how the operational processes of the system may impact the propagation of failures through the system, especially in cases where the direction of fault propagation along a specific hardware or software path may change with the operating mode. In addition there are failure modes that contain different failure effects in the upstream and downstream directions. For example, a valve failing closed will stop the flow in both the upstream and downstream directions, therefore a flow meter observation point could be located either before or after the valve and those effects need to be transferred in both directions. In addition, the deadheaded supply pressure would typically be high, while the pressure downstream of the failed valve would be low. In the FFM of this example, separation of backward (upstream) propagation effects and the forward (downstream) propagation effects is required to ensure proper modeling of the physical effects of the failure and to avoid the introduction of diagnostic errors.

Fifth, TEAMS FFMs allow for the assignment of both probability and criticality values to individual failure modes. The problem is that often these values can vary during the operational profile of the system. During diagnostic assessment it would be optimal to rank potential failure modes that are being detected by either or both of these variables. Techniques have been established to overcome some of this limitation during off-line design analyses, but development of a more flexible approach that would enable the use of these parameters in a real-time process has been evasive.

C. Implementation and Verification Testing

When the knowledge acquisition and conceptual design phases have progressed sufficiently, the implementation phase may begin. During the knowledge acquisition phase, information about the system design, the system operation, system failures, and system responses to failure were collected. Then, during the conceptual design phase, the requirements and architecture that will drive model implementation were defined. The next phase, Implementation and Verification, will focus on the model implementation itself and the review and testing required to verify the resulting FFM. For the process documented in this paper, implementation and model verification are typically performed in a tight iterative loop. This motivates the discussion of these two parts of the process under the same process phase.

1. Implementation

The development of FFMs has progressed from being an art form with no clear standards and practices to a reasonably proficient modeling technology with established guidelines. Still, individual modelers working independently will typically represent the failure space of a given subsystem or component using inconsistent nomenclature and modeling constructs. This can lead to incompatibilities, confusion, and verification issues during the integration of these models into a system-level FFM. Hence, it is important that individual modelers adhere to the modeling conventions and guidelines established during the conceptual design phase. Once the components to be modeled (breadth) and the hierarchical structure (depth) are defined, the FFM development team can begin the process of using TEAMS software to build the model.

TEAMS is the current software package used by NASA to develop FFMs. It provides a powerful GUI that allows the user to model FFMs as diagrams which visually represent the decomposed system. Using TEAMS, the user can create model elements to form the desired hierarchical structure and can define the effect propagation paths between the various elements. As the model develops, the user can also quickly move throughout the model hierarchical structure, search for modeling elements, and perform interactive testing of the model.

At this time, TEAMS does not provide a means to find user-specified information and replace it with alternate information throughout a model. The Batch Editor[17] is a stand-alone NASA-developed software tool that fills this void. This tool provides the user with a means to extract data from and make broad changes to a model. For example, using the Batch Editor, the user can query a FFM for the names of failure modes, review the resulting CSV (comma separated variable) file to identify and globally correct misspelled or incorrect names, and then use the Batch Editor to impose changes embodied in the revised CSV file on the FFM.

Likewise, TEAMS does not currently contain generic model libraries–although QSI is in the process of implementing a library capability. FFM developers should consider a development process that involves generic component libraries. Systems containing a significant number of similar or redundant components are good candidates for this approach. For these systems, the establishment of a generic component library offers a rigorous development structure for the initial FFM and a reduction in development time and resources for future FFMs that may be developed utilizing the resulting library. In addition, the generic component library offers a consistent representation of similar components which should enhance the accreditation process with the domain experts and users of the diagnostic assessments. NASA employed a generic component modeling process[21] under the Advanced Ground Systems Maintenance (AGSM) Integrated Health Management Demonstration for Universal Propellant Servicing System (IDU) project that could be used as a starting point for the development of other system-specific generic FFM component libraries.

The GEneric Model INstantIator (GEMINI) tool[21] for FFMs allows the modeler to develop a system-specific FFM component element. It does this by providing the modeler with a means of replacing fields in the generic component model with relevant information for a specific hardware or software component.

Figure 3 illustrates the generic component modeling process implemented during AGSM IDU project. Common components, such as valves, sensors and relief valves, were targeted as generic component candidates. A generic component library was established containing common component types, each of which were individually vetted and verified. The GEMINI tool then supported a consistent instantiation of the final set of component models which were again verified by reviews. The system-level hierarchical FFM was previously established with module placeholders that could be replaced with the instantiated component models. This process offered a systematic approach to FFM development and verification. When applying this process for the AGSM FFM development, a cost savings of 80% was demonstrated during the component modeling phase.

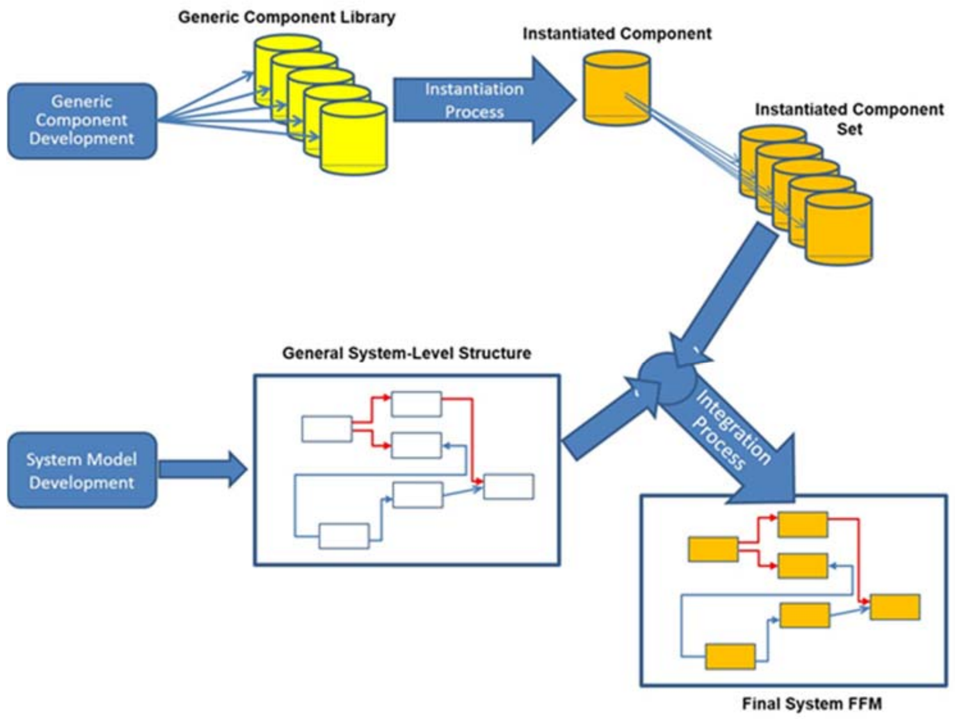

Generic Component Library
Instantiated Component
Generic Component Development
Instantiation Process
Instantiated Component Set
General System-Level Structure
System Model Development
Integration Process
Final System FFM

Figure 3. FFM development process incorporating the generic modeling process.

The Batch Editor and GEMINI are essential software tools that support model implementation. Future modelers would be wise to become familiar with and take appropriate advantage of each tool.

2. Verification Testing

Regardless of the FFM application, a series of test cases should be developed to verify that the FFM was built properly and performs to the expectations of the customer. In addition, these test cases should be retained for use during any regression testing of the FFM required in the future. For FFMs with a manageable number of failure modes, the testing can be exhaustive, including test cases for all possible failure mode assessments available from the model. For larger models, the selection of cases should be made to expose all model features and encompass all system configurations the FFM will be required to perform under.

The set of test cases should also include test cases that verify any FFM performance requirements. Each test case should be defined by the conditions and constraints of the test: the system mode or model configuration, the available tests, the active failure modes, and any interface processing data provided as input to the model in the case of real-time testing. The test case should also define the required output data or processed reports to be evaluated. Finally, the test case should define the expected or anticipated outcome and, thereby, define the success criteria for the test.

In addition to the internal model testing capabilities of the TEAMS software, three recently developed FFM verification tools are useful in significantly reducing model verification efforts: the D-matrix Comparator[18], the VERification Analysis (VERA) tool[20], and the Extended Testability Analysis (ETA) Tool[16].

A key output from TEAMS-based testability analysis is the Dependency Matrix or D-matrix. The D-matrix is a two-dimensional matrix that defines, for each model configuration, the complete detection coverage of the model. Figure 4 shows a portion of a D-matrix to illustrate this concept. The rows of the D-matrix are populated with the available or active failure modes for the analysis, while the columns contain all the active tests. Where row and column intersect, a one (1) value indicates that the test should detect the corresponding failure mode. The complete set of tests that detect a given failure mode is that failure mode's detection signature. Also as indicated in the figure, two or more failure modes which possess the same detection signature are therefore ambiguous with each other, meaning they cannot be distinguished. Changes to a given FFM are directly reflected in the associated D-matrix.

The D-Matrix Comparator is a NASA-developed software tool that provides a direct comparison between two D-Matrices generated by the different versions of the same FFM. This tool enables the rapid identification of the model

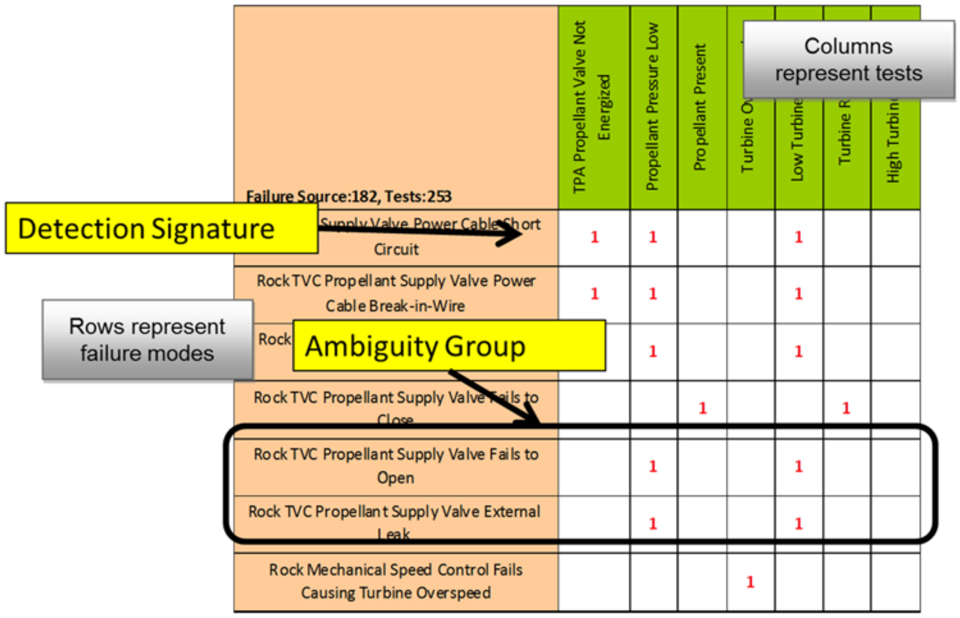

Failure Source:182, Tests:253
TPA Propellant Valve Not Energized
Propellant Pressure Low
Propellant Present
Turbine O
Low Turbine
Turbine R
High Turbin
Columns represent tests
Detection Signature
Supply Valve Power Cable Short Circuit
Rock TVC Propellant Supply Valve Power Cable Break-in-Wire
Rows represent failure modes
Rock
Ambiguity Group
Rock TVC Propellant Supply Valve Fails to Close
Rock TVC Propellant Supply Valve Fails to Open
Rock TVC Propellant Supply Valve External Leak
Rock Mechanical Speed Control Fails Causing Turbine Overspeed
1

Figure 4. Sample of an FFM Dependency Matrix

changes and the impact of those changes. This type of tool is fundamentally required for regression testing of the FFM.

The VERA Tool is a NASA-developed software tool that automates the inspection of the model for adherence to modeling conventions and practices described in Ref. 19. It is a Visual Basic application that analyzes the model and generates reports in Microsoft Excel. Reports generated by the VERA tool identify and report all violations of known modeling conventions and provides scores to quantify the success of the verification process and support model accreditation. It is strongly recommended that the VERA tool be used to analyze all FFMs and that violations be corrected prior to the integration of multiple FFMs and prior to the use of FFMs for analyses, studies, or monitoring an operating system.

The ETA Tool is a NASA-developed software tool that extends the analysis and reporting capabilities of TEAMS software and it can be used as follows to support the verification of FFMs. This software can be used to analyze an FFM and generate detailed reports for both model detectability and test utilization. The detectability report identifies failure modes that are *not* detected under the testability conditions embedded in the model, as well as those that *are* detected by the same conditions. The test utilization report identifies all the available tests that do *not* detect any failure modes, as well as a list of all failure modes that *are* detected by each test. It is recommended that these reports be reviewed with system designers and SMEs.

D. Application

The FFM is ready for use when available information about the system's failure space has been sufficiently captured by the FFM and the FFM has been verified and validated. In this section, currently available off-line analysis capabilities are described and the use of FFM in real-time diagnostic assessments is briefly discussed.

Seven analysis capabilities are available to the modeler through TEAMS software and through the ETA Tool: Failure Mode Detectability, Test Utilization, Failure Mode Isolation, Component Isolation, Failure Modes and Effects Analysis, Effect Mapping, and Sensor Sensitivity. Each of these analyses utilize the D-matrix, processing it in different ways to generate results that can significantly impact systems engineering and design processes. The first five analyses are available in both the TEAMS software and the ETA Tool. The last two analyses are only available in the ETA Tool. Instructions for using the ETA Tool and a complete description of the available analyses is currently available from the NASA Software Catalog as part of the ETA Tool v8.0 software release.

First, the Failure Mode Detectability analysis identifies failure modes that cannot be detected by tests available in the testability analysis test conditions. It then reports these failure modes while also reporting failure modes that can be detected and indicating those tests that detect the propagated failure mode effects. This is accomplished by reviewing each row of the D-matrix, extracting the detection signatures for each failure mode, and reporting all failure modes whose detection signature is empty. This analysis is particularly useful for verifying the level of detection provided by the available test suite for critical failure modes that lead to abort, caution & warning, or safing actions. A critical failure that cannot be identified by the modeled system design may indicate the need for a design change to incorporate additional sensors or logic.

Second, the Test Utilization analysis also processes information in the D-matrix, examining its columns rather than its rows. A test whose column in the D-Matrix is all zeroes indicates that the test is not useful for detecting any of the system failure modes. This may present any opportunity to simplify the system design through the elimination of detection hardware or software associated with the test. Here, prior to informing the systems engineering and design process, the modeler must take caution to ensure that all operating modes are analyzed. It would not be prudent to recommend the removal of a test that is unused in one mode if it is instrumental in detecting critical failures during other operating modes.

Third, the Failure Mode Isolation analysis examines the failure mode signatures for uniqueness or ambiguity. The root cause of failure modes with unique signatures can be determined unambiguously, while those without a unique signatures cannot. The report for this analysis lists both uniquely isolatable and ambiguous failure modes; organizing those that are ambiguous into groups that have common failure signatures. Under the NASA SLS program, this analysis was used to provide failure mode traces for specific monitored condition algorithms. To conduct the analysis, only tests defined in the algorithm of interest were incorporated into analysis. Results of the analysis reported failure modes that were detected by a sufficient number of algorithms to indicate detection. SMEs then reviewed the results to verify algorithm performance.

Fourth, the Component Failure Isolation analysis is similar to the Failure Mode Isolation analysis. This analysis determines if the various failure modes can be isolated to components pre-specified by the modeler. The report for this analysis groups failure modes based on common detection signatures. These failure mode isolation groups are further divided into sub-groups based on user-specified components. This analysis supports the verification of fault isolation requirements related to maintenance processes and logistics support. One example is a line replaceable unit (LRU) – a component that must be replaceable while a spacecraft is on the launch pad. To facilitate quick replacement, LRU designs typically require that all failure modes originating from the LRU be unambiguously isolatable to the LRU.

Fifth, the FMEA reporting capability has been designed to query the model and generate a FMEA report for the modeled system that is similar in content to typical NASA FMEA reports. Since a FFM contains all of the critical failure modes of the system and the tests available to detect them, it is essentially an “actionable” FMEA. As has been shown, the FMEA embodied by the FFM can be used to conduct a variety of analyses useful to systems engineering and design processes. These same analyses are tedious and difficult, if not impossible, to perform with a standard FMEA report. On the other hand, as the FFM is updated to reflect changes in the system design, the FMEA reporting capability can be used with the FFM to provide a “snap shot” of the failure modes and effects. Further, if the FFM has been developed strictly to meet system diagnostic requirements, the model, and consequently the resulting FMEA report, will contain only the detail required to meet the those requirements.

Sixth, the Effect Mapping Report processes the D-Matrix to locate tests attached to pseudo test points. Pseudo test points are intended to represent system level effects that do not necessarily correspond to specific sensor measurements. An example of this is loss of component or subsystem redundancy. This analysis supports an assessment of the effect of failures on the system, e.g., vehicle-level loss of mission analysis and probability risk assessment. It can also support the verification of detection and/or isolation of various system-level conditions, including conditions that lead to loss of mission or loss of redundancy.

Seventh, the Sensor Sensitivity analysis examines the impact of losing a sensor or sensors on the system’s ability to meet diagnostic requirements. It does this by systematically removing individual sensors or groups of sensors and their associated tests from the D-matrix before computing diagnostic performance metrics. This analysis can be used to confirm diagnostic performance in the presence of sensor failures (e.g. requirements of single fault tolerance).

As noted previously, FFMs are also capable of being used in a real-time mode. For a real-time application, the FFM tests are aligned to data processing functions that interface the FFM with a data acquisition system. This coordination between model and interface software is crucial and facilitated with the implementation of modeling conventions and the proper establishment of requirements for not only the FFM, but also for the interface software and the monitored system. For example, the FFM will require explicit input from the monitored system as to its operational phase in order to set the model configuration. Under real-time processing, the FFM is presented with tests that failed during the current time interval. The FFM identifies the failure mode or modes that could have generated the effects associated with that detection signature. The FFM can then provide information from the failure modes, as well as the location of the failures, to the operators in real-time.

Under the AGSM project, a FFM model was developed that would have supported an integrated health management demonstration. Following a process similar to that outlined in this paper, development of the FFM utilized generic component models and strict modeling conventions. The alignment of tests defined in the FFM with the interface software was carried out utilizing a Test Description File (TDF). The TDF aligns the test in the model with the test algorithms in the interface, defining input measurements and applied operational phase-based thresholds. Real-world data collection issues, such as data update rates and data availability, imposed requirements on the software interface to provide checks that ensured proper test results. In addition, the ability of the real-time FFM software package to continue providing diagnostic assessments, in the face of data loss due to sensor failure or intermittent loss of communication, required establishment of policy acceptable to consumers of the FFM output. A NASA-developed software tool called the Test Description File Creator (TDF Creator) was developed to streamline the development of the TDF for a particular application. Due to budget challenges, the AGSM-funded demonstration did not reach completion. However, preparation for the demonstration was a valuable exercise to advance and reinforce key areas of the overall FFM development process, particularly for the real-time aspect.

III. Concluding Remarks

This paper presented a process to support the development of Functional Fault Models (FFMs). The process documented has proved beneficial to recent systems engineering assessments supporting the design and operation of large systems under NASA (National Aeronautics and Astronautics) Ares I, Space Launch System, and Ground systems Development and Operations programs. The process is described in four phases. For each phase special emphasis was placed on key approaches, capabilities, and tools that are unique to FFMs. The first phase focused on knowledge acquisition and included a discussion of system diagnostic requirements that drive the process, as well as, FFM-relevant system design data and modeling knowledge. The second phase focused on the conceptual design, delving into the definition of requirements for an FFM, the development of a conceptual architecture for an FFM, and some limitations of FFMs. The third phase focused on implementation and verification testing with a discussion of the recent tools developed to support this phase. The fourth, and final, phase described the application of analysis capabilities to specific systems engineering assessments and how the results of those assessments can feedback into and benefit ongoing systems engineering and design processes.

It is important to reemphasize that the FFM development process reported here is intended to be an iterative process that evolves the model and feeds back into systems engineering and design processes as the system design evolves over time. One phase of the process need not be fully mature before progressing to the next. It is the qualitative nature of these models combined with the early adoption by the systems development community that will result in FFMs providing the largest benefit to the resulting system design.

IV. Acknowledgments

The authors wish to acknowledge the support of the Mission and Fault Management (M&FM) project under NASA’s Space Launch System Program and the Advanced Ground Systems Maintenance (AGSM) project under NASA Ground Systems Development and Operations program for their support in advancing FFM capabilities. The M&FM project is led by Marshall Space Flight Center and the AGSM Project is led by Kennedy Space Center.

V. References

[1]Kurtoglu, T., and Tumer, I.Y., “A Graph-Based Fault Identification and Propagation Framework for Functional Design of Complex Systems,” *ASME Journal of Mechanical Design*, Vol. 130, No. 051401, May 2008, pp. 1-8.

[2]Padalkar, S., Karai, G., and Sztipanovits, J., "Graph-based Real-time Fault Diagnostics," Fourth Conference on Artificial Intelligence for Space Applications, October 1988, pp. 115-123.

[3]Pattapati, K., and Alexandridis, M.G., "Application of Heuristic Search and Information Theory to Sequential Fault Diagnosis," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 20, No. 4, July/August 1990, pp. 872-887.

[4]Pattipati, K.R., Raghavan, V., Shakeri, M., Deb, S., and Shrestha, R., “TEAMS: Testability Engineering and Maintenance System,” *American Control Conference*, Baltimore, MD, June 1994, pp. 1989–1996.

[5]Deb, S., Pattipati, K.R., Raghavan, V., Shakeri, M. and Shrestha, R., “Multi-Signal Flow Graphs: A Novel Approach for System Testability Analysis and Fault Diagnosis,” *IEEE Automatic Testing Conference*, Anaheim, CA, September 20-22, 1994.

[6]Patterson-Hine, F.A., Davis, G.J., and Pedar, A., "The Role of Reliability Graph Models in Assuring Dependable Operation of Complex Hardware/Software Systems," AIAA 91-3793, *AIAA 8th Computing in Aerospace Conference*, Baltimore, MD, October 21-24, 1991.

[7]Shakeri, M., Pattipati, K., Raghavan, V., Patterson-Hine, A., and Kell, T., "Sequential Test Strategies for Multiple Fault Isolation," *IEEE AUTOTESTCON*, Atlanta, GA, August 1995.

[8]Shakeri, M., Raghavan, V., Pattipati, K., and Patterson-Hine, A., "Sequential Testing Algorithms for Multiple Fault Isolation," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 30, No. 1, January 1996, pp. 1-14.

[9]Iverson, D.L., Patterson-Hine, F.A., "Digraph Reliability Model Processing Advances and Applications," AIAA-93-4677-CP, *9th Computing in Aerospace Conference*, San Diego, CA, October 19-21, 1993.

[10]Patterson-Hine, A., Hindson, W., Sanderfer, D., Deb, S., Domagala, C., "A Model-based Health Monitoring and Diagnostic System for the UH-60 Helicopter," *American Helicopter Society International 57th Annual Forum and Technology Display*, Washington, DC, May 9-11, 2001.

[11]Deb, S., Ghoshal, S., Malepati, V., Domagala, C., Patterson-Hine, A., and Alena, R., "Remote Diagnosis of the International Space Station Utilizing Telemetry Data,*" SPIE AeroSense Conference*, Orlando, FL, April 16-19, 2001.

[12]Deb, S., Domagala, C., Shrestha, R., Malepati, V., Cavanaugh, K., Patterson-Hine, A., Sanderfer, D., and Cockrell, J., "Model-based Testability Assessment and Directed Troubleshooting Of Shuttle Wiring Systems," *SPIE AeroSense Conference*, Orlando, FL, April 16-19, 2001.

[13]Schwabacher, M.A., Martin, R.A., Waterman, R.D., Oostdyk, R.L., Ossenfort, J.P., and Matthews, B., "Ares I-X Ground Diagnostic Prototype," *AIAA Infotech@Aerospace Conference*, Atlanta, GA, April 20-22, 2010.

[14]Ferrell, B., Lewis, M., Perotti, J, Oostdyk, R., Spirkovsha, L., Hall, D., and Brown, B., "Usage of Fault Detection Isolation & Recovery (FDIR) in Constellation (CxP) Launch Operations,", AIAA 2010-2181, *AIAA SpaceOps Conference*, Huntsville, AL, April 25-30, 2010.

[15]Maul, W., Melcher, K., Chicatelli, A. and Johnson, S., "Application of Diagnostic Analysis Tools to the Ares I Thrust Vector Control System," *AIAA InfoTech@Aerospace Conference*, Atlanta, Ga, 2010.

[16]Maul, W.A. and Fulton, C.E., "Extended Testability Analysis Tool User Guide," AIAA 2011-1637, *AIAA InfoTech@Aerospace Conference*, St. Louis, Missouri, March 29-31, 2011.

[17]Barszcz, E., Robinson, P., and Fulton, C., "Tools Supporting Development and Integration of TEAMS Diagnostic Models," AIAA 2011-1639, *AIAA InfoTech@Aerospace Conference*, St. Louis, Missouri, March 29-31, 2011.

[18]Aaseng, G.B., Barszcz, E., Valdez, H., and Moses, H., Scaling Up Model-Based Diagnostic and Fault Effects Reasoning for Spacecraft," *AIAA SPACE Conferences and Exhibition*, AIAA 2015-4465, Pasadena, CA, August 31-September 2, 2015.

[19]K0000190582-GEN, "Testability Engineering and Maintenance System (TEAMS) Modeling Conventions and Practices," National Aeronautics and Space Administration, 2014.

[20]Bis, R., and Maul, W., "Verification of Functional Fault Models and the Use of Resource Efficient Verification Tools," AIAA 2015-1796, *AIAA InfoTech@Aerospace Conference*, Kissimmee, Florida, January 5-9, 2015.

[21]Maul, W.A, Hemminger, J.A., Oostdyk, R, and Bis, R.A., "A Generic Modeling Process to Support Functional Fault Model Development," AIAA 2016-2134, *AIAA Infotech@Aerospace Conference*, San Diego, CA, January 4-8, 2016.